HARSH KASLIWAL

2003085

C21

**ASSIGNMENT 1**

1. Python program to implement operations of linked list

1. Display list
2. Insert at beginning
3. Insert at End
4. Insert at specified postion
5. Delete from beginning
6. Delete from end
7. Delete at specified postion
8. Delete a particular element
9. Search an element
10. Replace element at specified index
11. Forward traversal
12. Reverse traversal
13. Insert after an element

**CODE:**

class Node: def \_\_init\_\_(self,value,next): self.value = value self.next = next

class Linked\_list:

def \_\_init\_\_(self):

self.head = None

def insert\_at\_beginning(self,value):

self.head = Node(value,self.head)

def insert\_at\_end(self,value): if self.head == None:

self.head = Node(value,None)

i = self.head

while(i.next): i = i.next

i.next = Node(value,None)

def insert\_at\_specified\_position(self,value,pos): if pos == 1:

node = Node(value,self.head) self.head = node return

i = self.head while(pos != 2): pos = pos - 1 i = i.next

node = Node(value,None) node.next = i.next i.next = node

def delete\_from\_beginning(self):

self.head = self.head.next

def delete\_from\_end(self): i = self.head while(i.next != None):

prev = i i = i.next prev.next = None

del(i)

def delete\_at\_specified\_position(self,pos):

ptr\_del = self.head prev = self.head i = self.head.next

while(pos != 1): pos -= 1 prev = ptr\_del ptr\_del = i i = i.next prev.next = i del(ptr\_del) def delete\_particular\_element(self,val):

i = self.head ptr\_del = self.head prev = self.head while(ptr\_del.value != val): prev = ptr\_del ptr\_del = i i = i.next prev.next = i del(ptr\_del) def search\_element(self,val): i = self.head count = 1 while(i.next != None): if(i.value == val):

print("Element is Present at position",count) return

else:

count += 1 i = i.next

print("Element not present in linked list") def replace\_element(self,pos,val): i = self.head

if pos == 1:

i.value = val return

while(pos != 1): pos = pos – 1

i = i.next i.value = val def reverse\_traversal(self):

i = self.head prev = self.head lst = [] while(i != None):

lst.append(i.value)

i = i.next

lst.reverse() print("Reverse Traversal is : ")

for i in lst: print(i,"-->",end = "")

print()

#same as forward traversal def display\_list(self): if self.head == None:

print("Linked list is Empty!!!") return i = self.head linked\_list = '' while(i): linked\_list += str(i.value) + '-->' i = i.next print(linked\_list)

def insert\_after\_element(self,val,ele):

i = self.head while(i.next != None): if(i.value == val):

i.value = ele print("value changed") return i = i.next print("No such element found")

if \_\_name\_\_ == '\_\_main\_\_':

ll = Linked\_list()

print("Inserting elements in the beginning") ll.insert\_at\_beginning(10) ll.insert\_at\_beginning(200) ll.insert\_at\_beginning(18) ll.insert\_at\_beginning(30) ll.display\_list() print()

print("Inserting elements at the end") ll.insert\_at\_end(20) ll.insert\_at\_end(40) ll.insert\_at\_end(50) ll.insert\_at\_end(60) ll.display\_list() print()

print("Inserting element 100 at position 5") ll.insert\_at\_specified\_position(100,5) ll.display\_list() print()

print("Deleting the first element") ll.delete\_from\_beginning() ll.display\_list() print()

print("Deleting last element") ll.delete\_from\_end() ll.display\_list() print()

print("Deleting element from position 3") ll.delete\_at\_specified\_position(3) ll.display\_list() print()

print("Deleting element 20") ll.delete\_particular\_element(20) ll.display\_list() print()

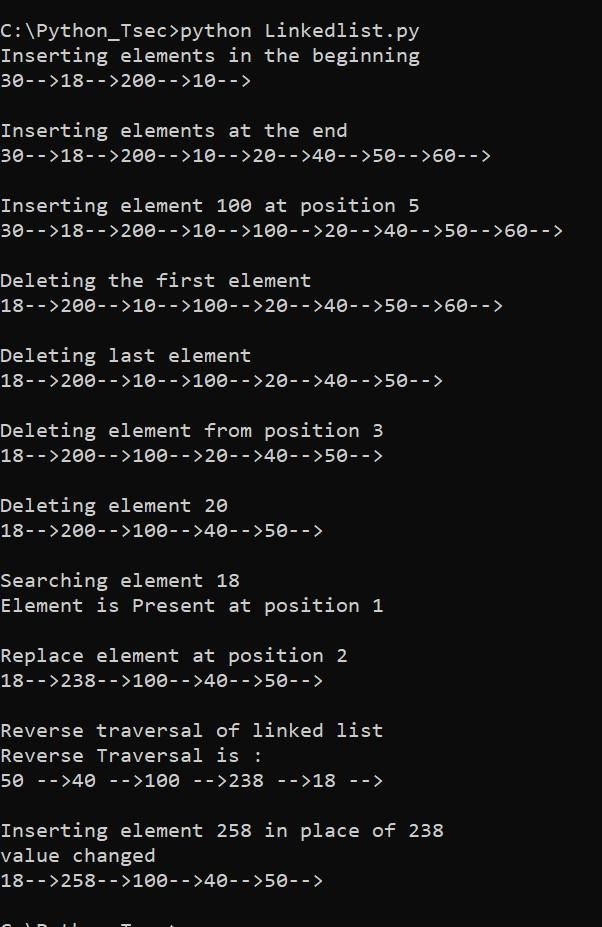
print("Searching element 18") ll.search\_element(18) print()

print("Replace element at position 2") ll.replace\_element(2,238) ll.display\_list() print()

print("Reverse traversal of linked list") ll.reverse\_traversal() print()

print("Inserting element 258 in place of 238") ll.insert\_after\_element(238,258) ll.display\_list()

**Output:**



2. Python program to implement Stack data structure create class Stack with following functions.

* Push an element
* Pop an element
* Top of Stack
* Search an element
* Display stack create instance and perform all operations

**Code:**

#stack.py class Stack:

def \_\_init\_\_(self): self.st = [] def isempty(self):

return self.st == []

def push(self,element):

self.st.append(element)

def pop(self): if self.isempty(): return -1

else:

return self.st.pop()

def peek(self):

n = len(self.st) return self.st[n-1] def search(self,element): if self.isempty():

return -1 else:

try:

n = self.st.index(element) return len(self.st)-n except ValueError:

return -2 def display(self): return self.st

# Main Code: from stack import Stack s = Stack() choice = 0 while choice<6: print('STACK OPERATIONS') print('1.Push element') print('2.Pop element') print('3.Top of stack') print('4.Search an element') print('5.Display Stack') print('6.Exit') choice = int(input('Your choice: ')) if choice==1: element = int(input('Enter element: ')) s.push(element) elif choice==2: element = s.pop() if element == -1:

print('The stack is empty') else:

print('Popped element= ', element) elif choice==3:

element = s.peek() print('Topmost element= ', element) elif choice==4:

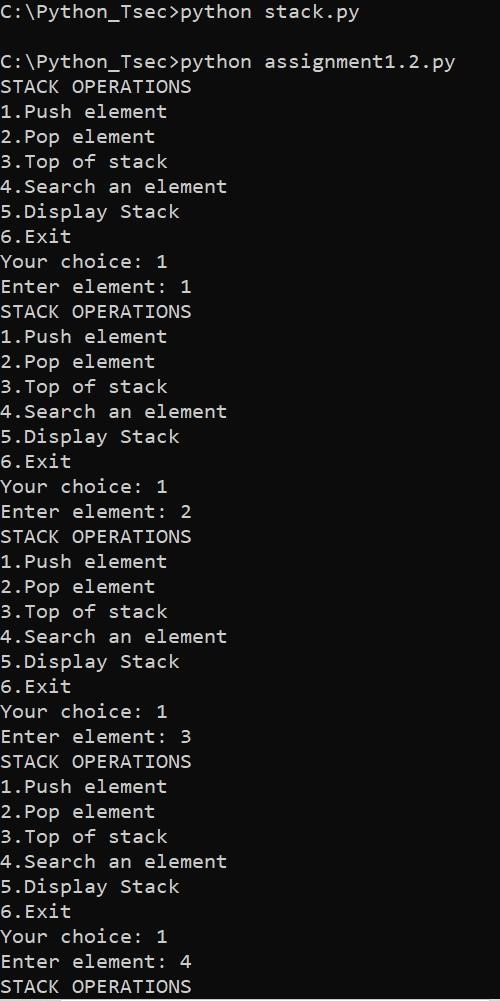
element = int(input('Enter element: ')) pos = s.search(element) if pos == -1:

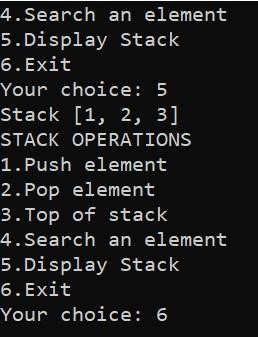
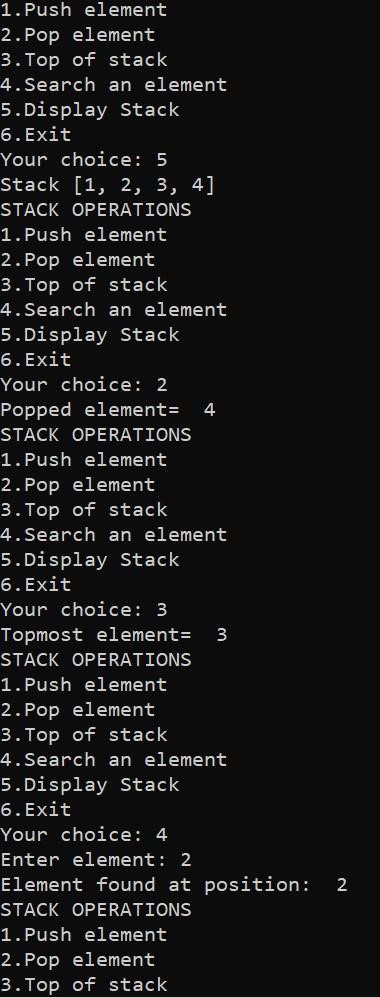
print('The stack is empty') elif pos == -2:

print('Element not found in the stack') else:

print('Element found at position: ', pos) elif choice==5:

print('Stack',s.display()) else: break

**Output:**



3. Python program to implement Queue data structure create class Queue with following functions.

* Insert an element
* Remove an element
* Search an element
* Display queue

**Code:**

#que1.py class Queue:

def \_\_init\_\_(self):

self.qu = [] def isempty(self): return self.qu == []

def add(self,element):

self.qu.append(element) def delete(self): if self.isempty(): return -1

else:

return self.qu.pop(0) def search(self,element): if self.isempty(): return -1 else: try:

n = self.qu.index(element) return n+1

except ValueError: return -2

def display(self):

return self.qu # Main

Code: from que1 import Queue q = Queue() choice = 0 while choice<5:

print('QUEUE OPERATIONS') print('1.Insert an element') print('2.Remove an element') print('3.Search an element') print('4.Display Queue') print('5.Exit')

choice = int(input('Your choice: ')) if choice==1: element = int(input('Enter element: ')) q.add(element) elif choice==2:

element = q.delete() if element == -1:

print('The queue is empty') else:

print('Removed element= ', element) elif choice==3:

element = int(input('Enter element: ')) pos = q.search(element) if pos == -1: print('The queue is empty') elif pos == -2:

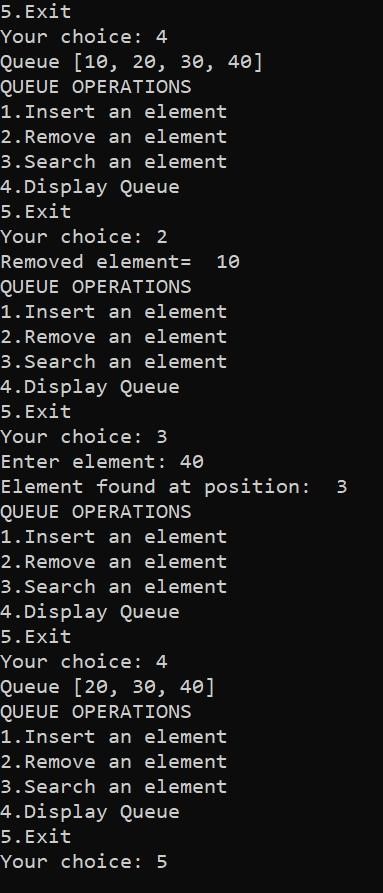
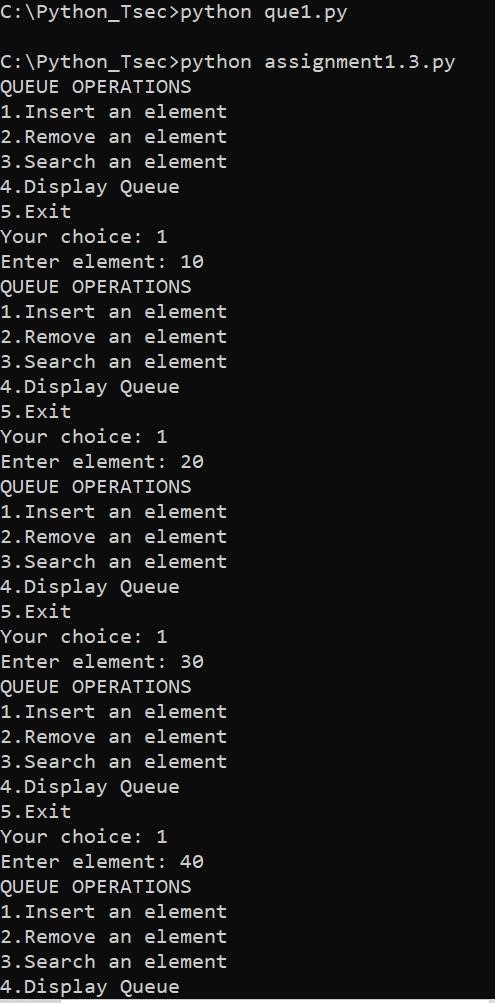
print('Element not found in the queue') else:

print('Element found at position: ', pos) elif choice==4:

print('Queue',q.display())

else: break

**Output:**



4. Python program to use deque class from collections with following functions.

* Add element at Front
* Remove element from Front
* Add element at Rear ● Remove element from Rear
* Search for an element

**Code:**

from collections import deque d = deque() choice = 0 while choice<6:

print('DEQUE OPERATIONS')

print('1.Insert an element at front') print('2.Remove

an element from front') print('3.Insert an element at rear') print('4.Remove an element from rear') print('5.Search an element')

print('6.Exit') choice = int(input('Your choice: ')) if choice==1:

element = int(input('Enter element: ')) d.appendleft(element) elif choice==2:

if len(d) == 0:

print('Deque is empty') else:

d.popleft() elif choice==3:

element = int(input('Enter element: ')) d.append(element) elif choice==4:

if len(d) == 0:

print('Deque is empty') else:

d.pop() elif choice==5:

element = int(input('Enter element: ')) c = d.count(element) print('No of times the element found:', c) else: break print('Deque=' , end='') for i in d: print(i, '

', end='') print()

**Output:**

